Virtualized System Development

Using hardware simulation
to help software development

Dr. Jakob Engblom
Virtutech

Virtutech

- Virtutech develops virtual platforms
  - Both standard virtual boards and customer-specific platforms
- Based on full-system simulation technology
- Flagship product: Simics 4.0
- We help our customers
  - Save money
  - Develop better products with less risk
  - Get products to market sooner

Virtualized System Development

Virtuali-what?

Virtualization

- Making a computer program behave like a computer for the purpose of running software
- Mechanisms for making some computer resource less subject to physical constraints
- Virtual memory, disk virtualization, virtual machines, …
- Very hip in the data center world currently

Virtutech Simics is really doing a bit of all of these, and it has been called all three

Simulation

- A piece of software that simulates something
- Used to perform experiments and gain insight not possible in the real world
- Control and insight into the internals of the process big advantages
- For computing, often associated with being slow
- Physics, computers, weather, games, …

Emulation

- A piece of software that mimics some computer software or hardware
- Focused on the execution of existing software
- Terminal emulators, OS emulators, console emulators

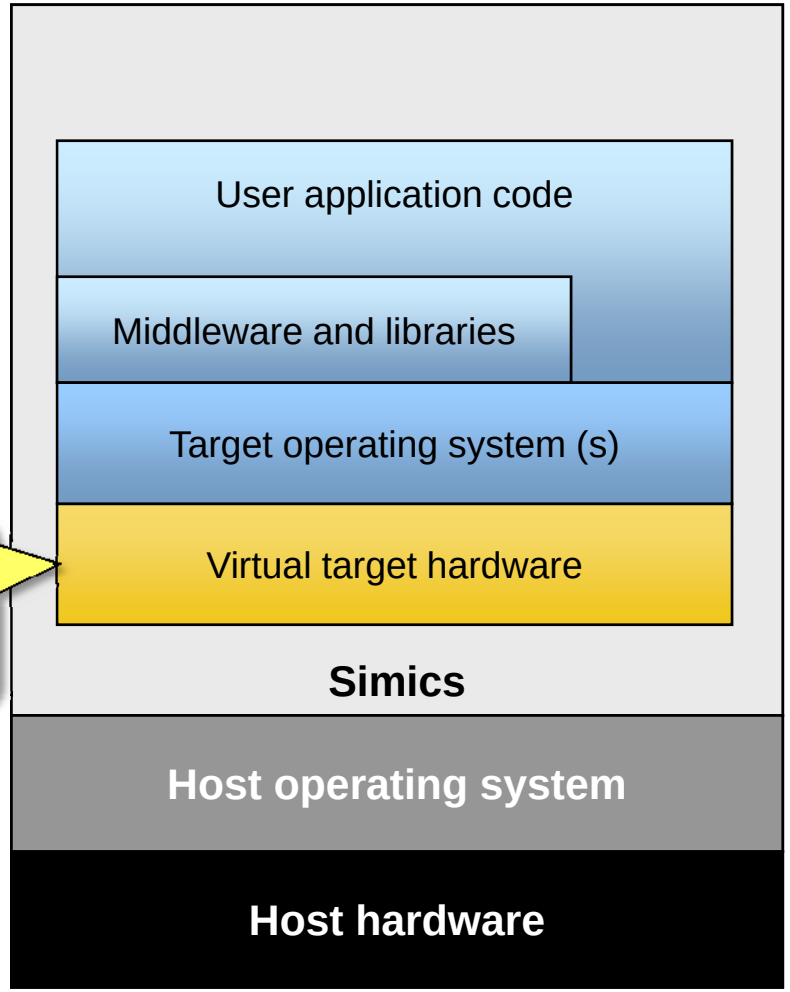
virtutech

What is Virtual Hardware?

- A piece of software
- Running on a regular PC, server, or workstation
- Functionally identical a particular hardware
- Runs the same software as the physical hardware system

Virtutech Core Technology

- Model any electronic system on a PC or workstation
  - Simics is a software program, no hardware required
- Run the exact same software as the physical target (complete binary)
- Run it fast (100s of MIPS)
- Model any target system
  - Networks, SoCs, boards, ASICs, ... no limits
- For the benefit o
  developers and
  providers
- Enables process change in software development



Typically, an embedded or real-time control computer system

Why use Virtual Systems?

“Because hardware is no fun”

Because Hardware Is...

Not yet available

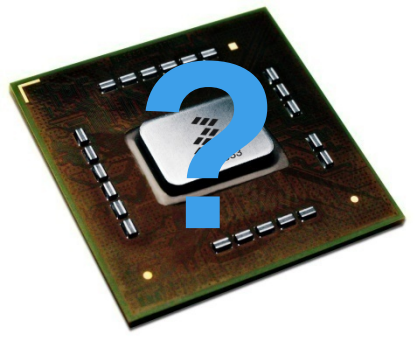
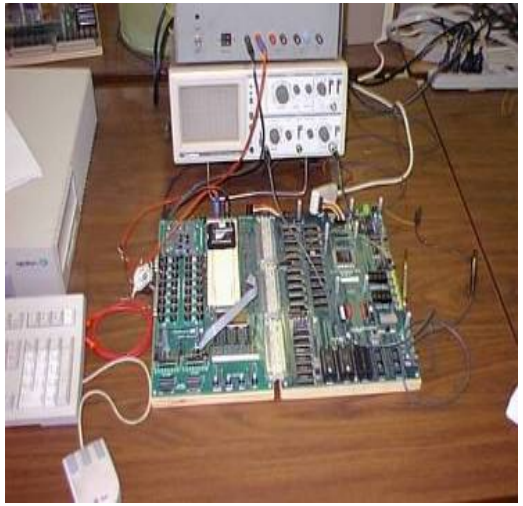


Photo: Freescale

Flaky prototype stage



Not available anymore

Photo: Computer History Museum

Because Hardware Is...

Inconvenient

Dangerous

Photo: www.mil.se, Bromma Conquip

Inaccessible

Photo: ESA

Because Hardware Is...

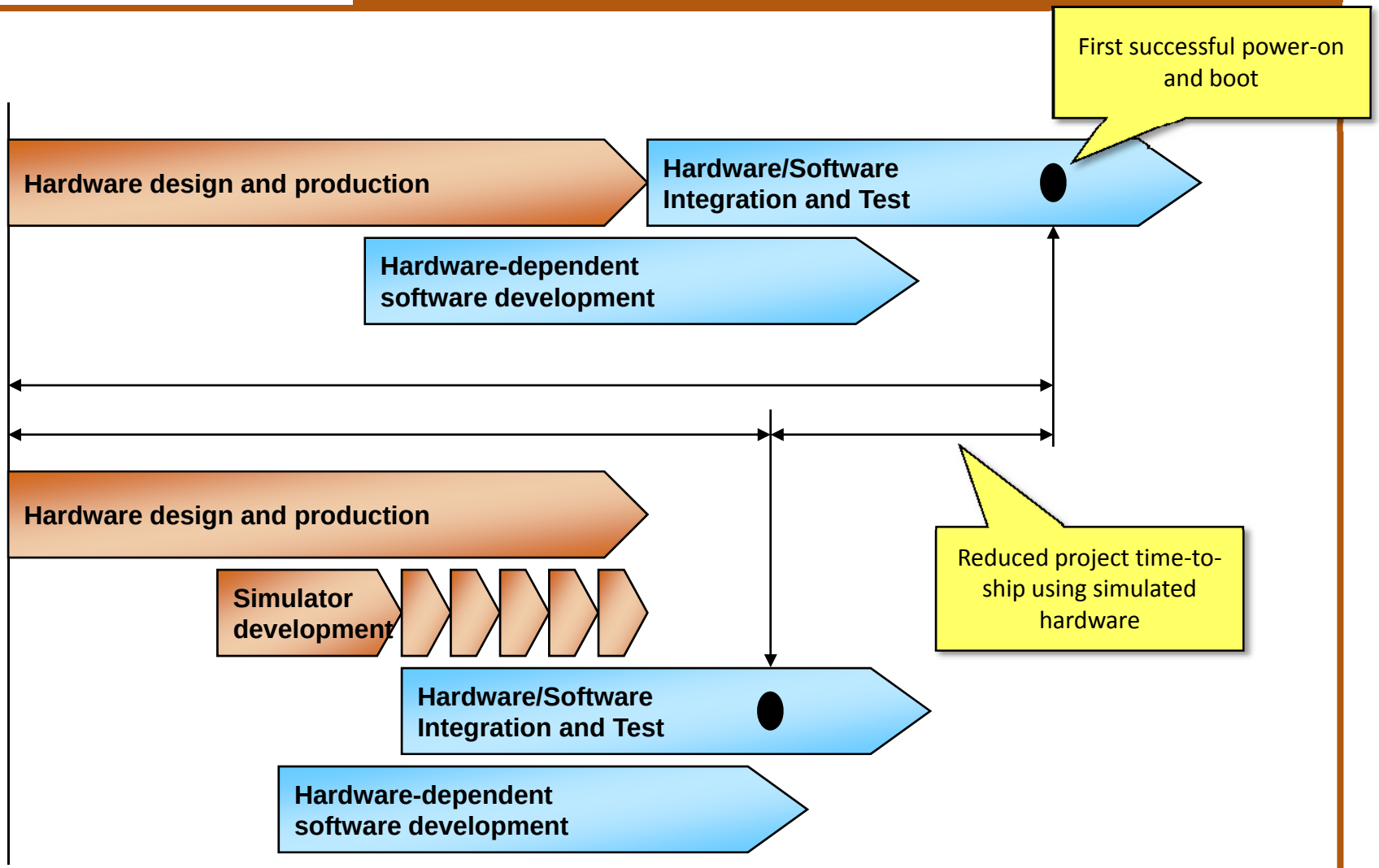
Impractical in scale

Limited

Inflexible

Virtual Platform Advantages and Features

Example: Early Hardware



virtutech

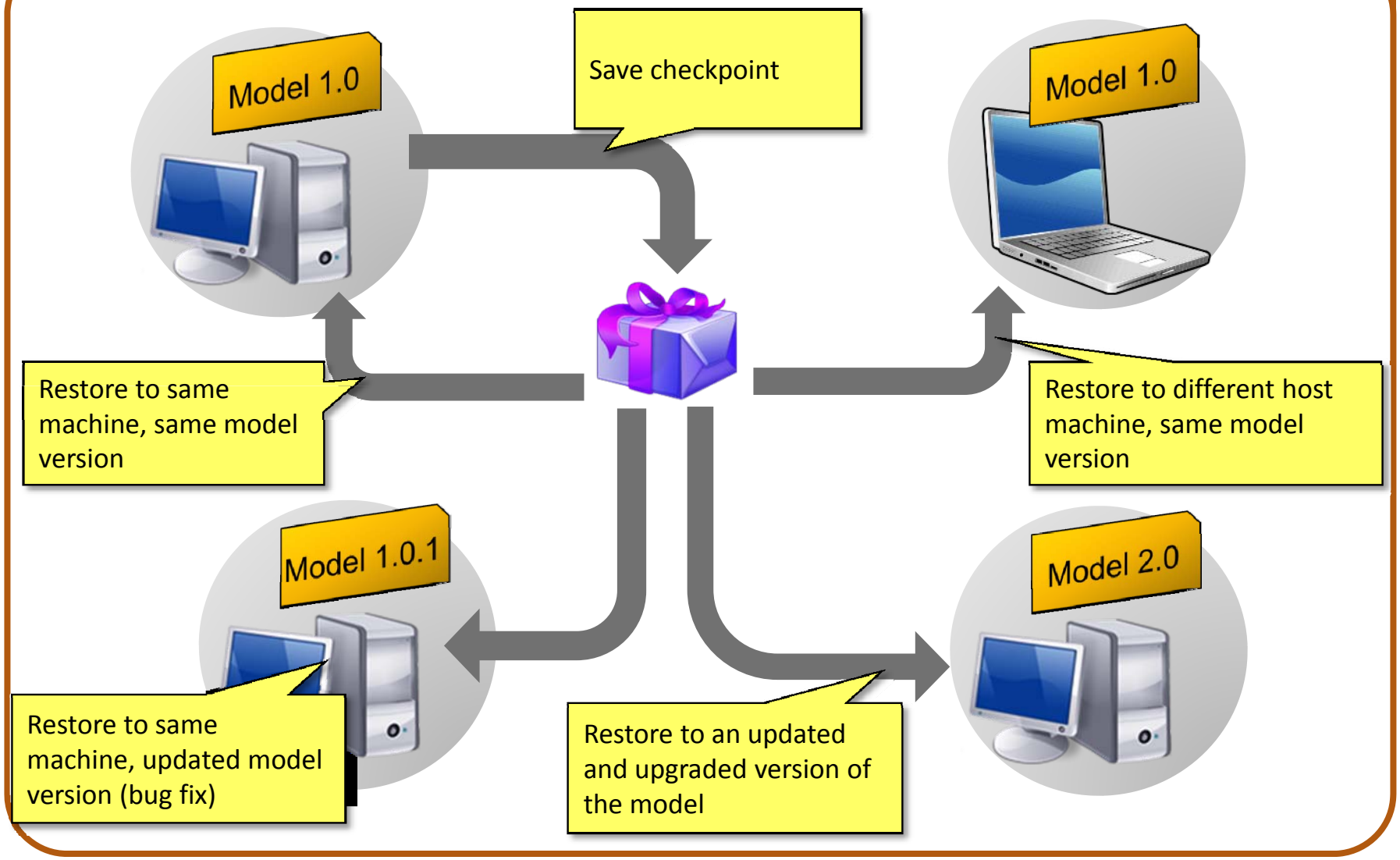
Handy Features of Simulation

- Checkpointing
  - Store current state; pick up and continue later
  - Position workload once, use many times
  - Spread a system state to multiple developers
  - Package error reports
  - Can checkpoint an entire network of machines
  - Key for repeating executions

- Determinism/Repeatability
  - Same initial state gives same execution;
  - Repeat the same execution any number of times
  - Investigate a problem time after time
  - For multiprocessor systems & network systems
  - Very useful for complex systems, where repeatable runs otherwise do not happen

virtutech

Checkpointing in Simics

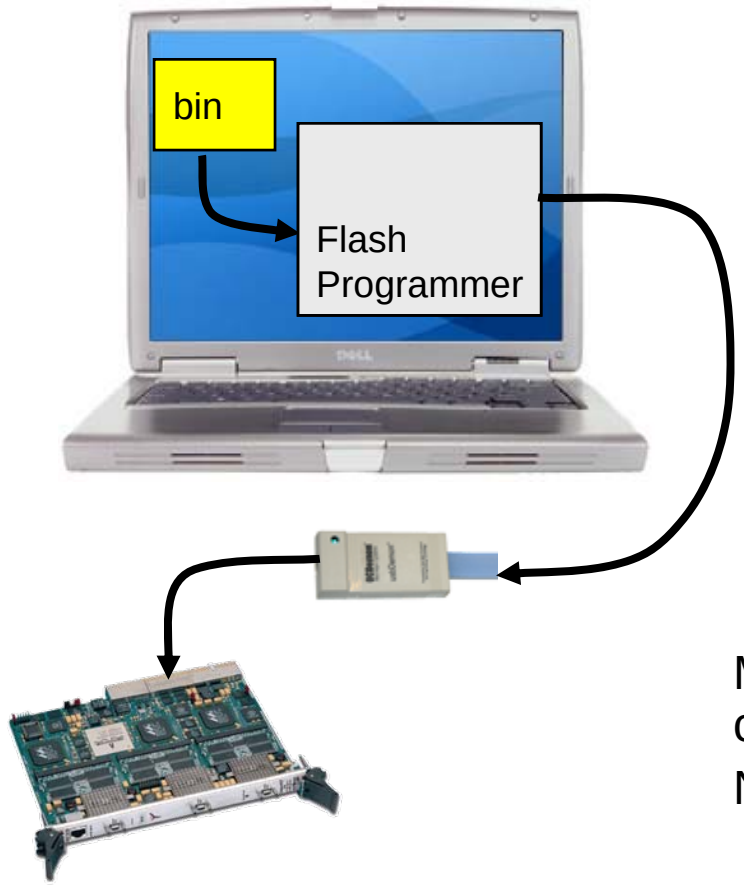


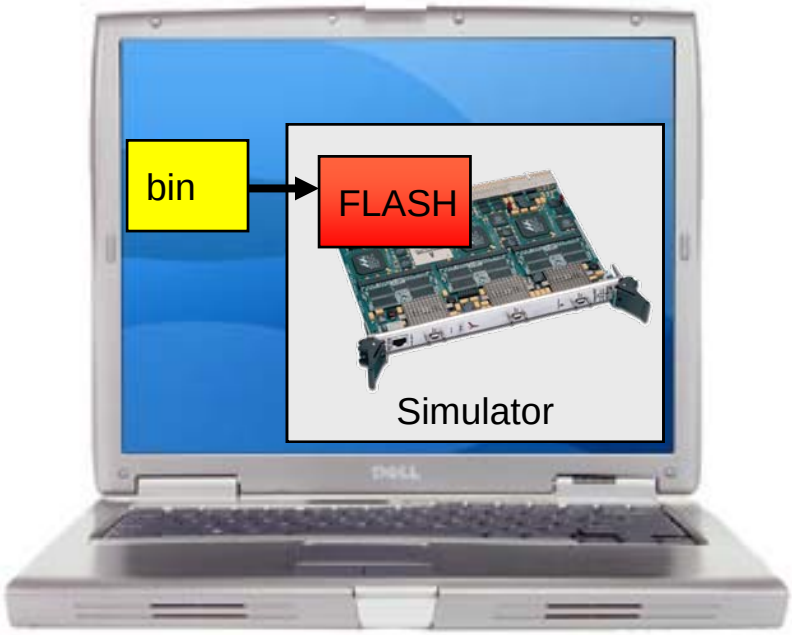
virtutech

Handy Features of Simulation

- Visibility (insight without intrusion)
  - All state can be observed
  - All events can be traced and logged
- Controllability
  - Any part of machine or state can be changed
  - Fault injection
- Virtual time
  - Time is completely virtual
  - Global synchronization across all machines in a network
  - Global stop across all processors in a multiprocessor

Convenience: Loading Flash





Much shorter turn-around time for changing target software setup

No risk of “bricking” a target

virtutech

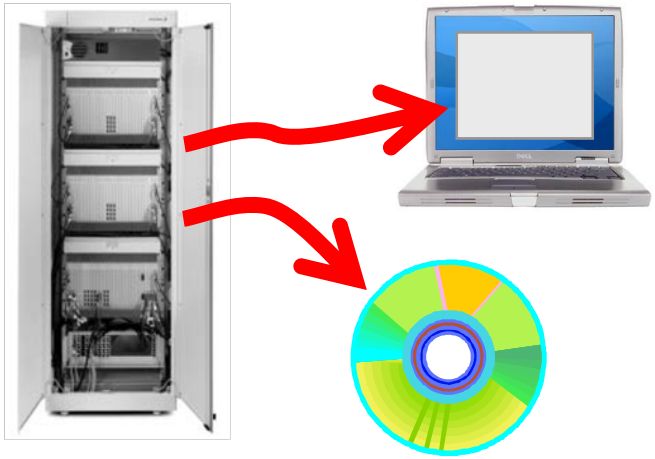
Handy Features of Simulation

- Configurability
  - Any parameter of system can be changed
- Sandboxing
  - Allows investigating ”nasty code”
  - Simulated machine complete isolated
  - Networks can be isolated
  - Simics undetectable by malware
    - Complete hardware simulation, no virtualization tricks
- Reverse execution
  - Roll back execution to previous state
  - Reverse breakpoints
  - Investigate details of program errors

Hardware Availability

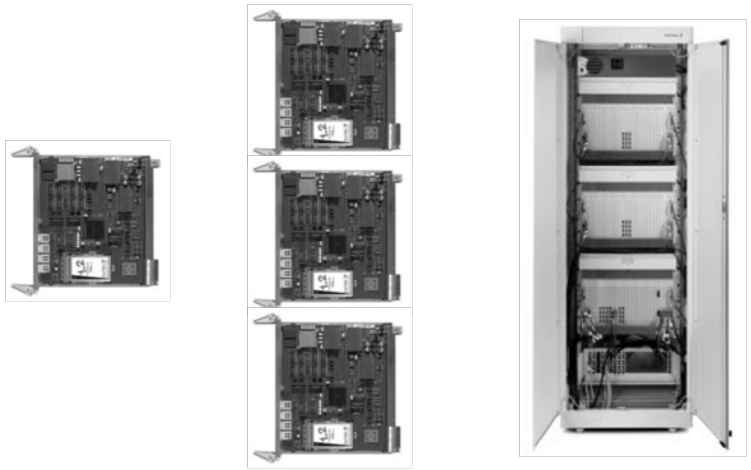
Wide Availability

- Virtual system is ”just software”
- Trivial to copy
- Trivial to distribute
- Each engineer can have a custom hardware system at their desk

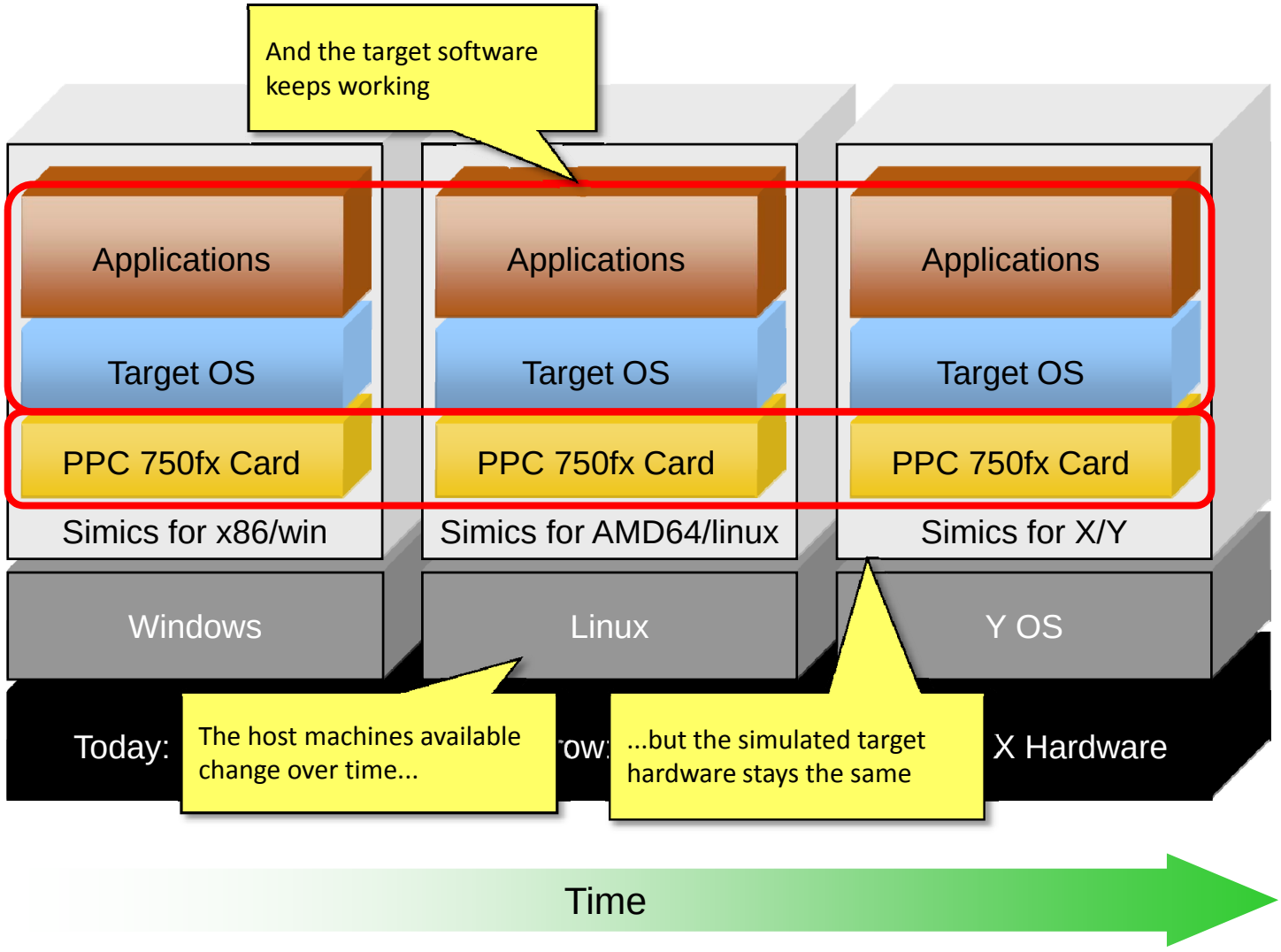


Scalable

- No physical supply limit
  - Any number of each type of board
  - Any type of system in ”infinite” supply
- A virtual system can be big or small by simple software (re)configuration

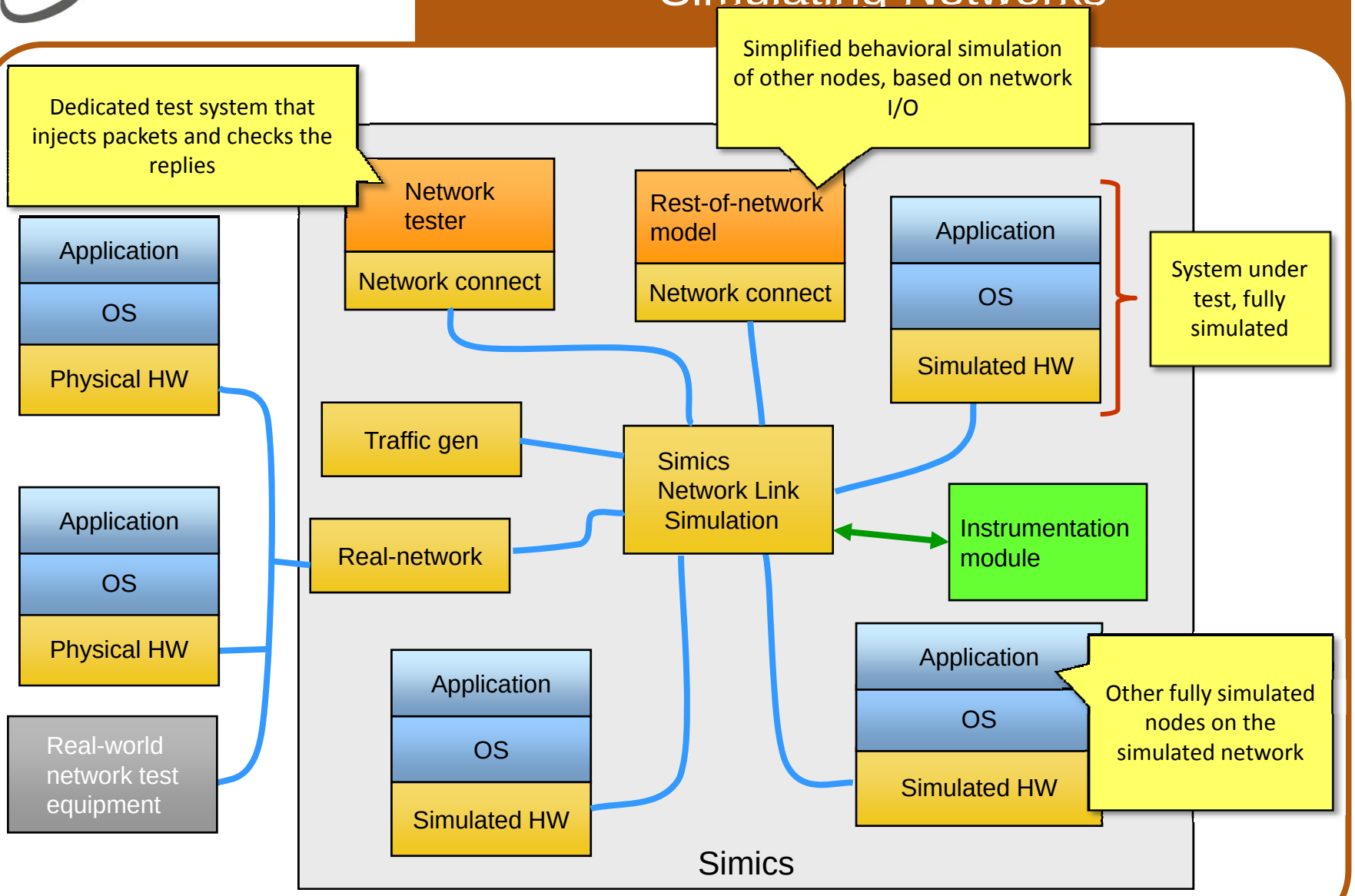


Virtualization = Infinite Longevity



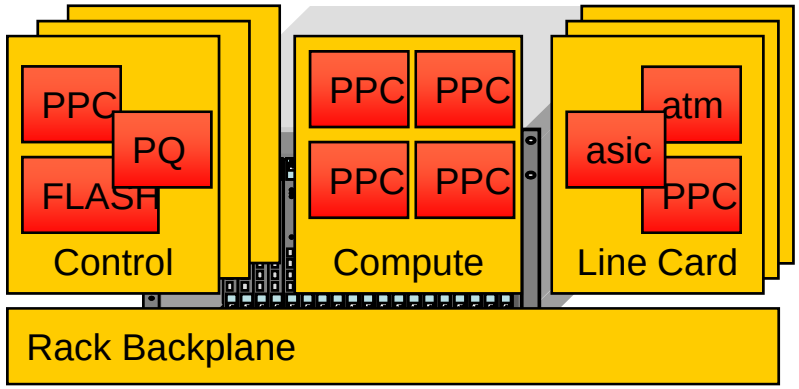
Example Systems

Simulating Networks



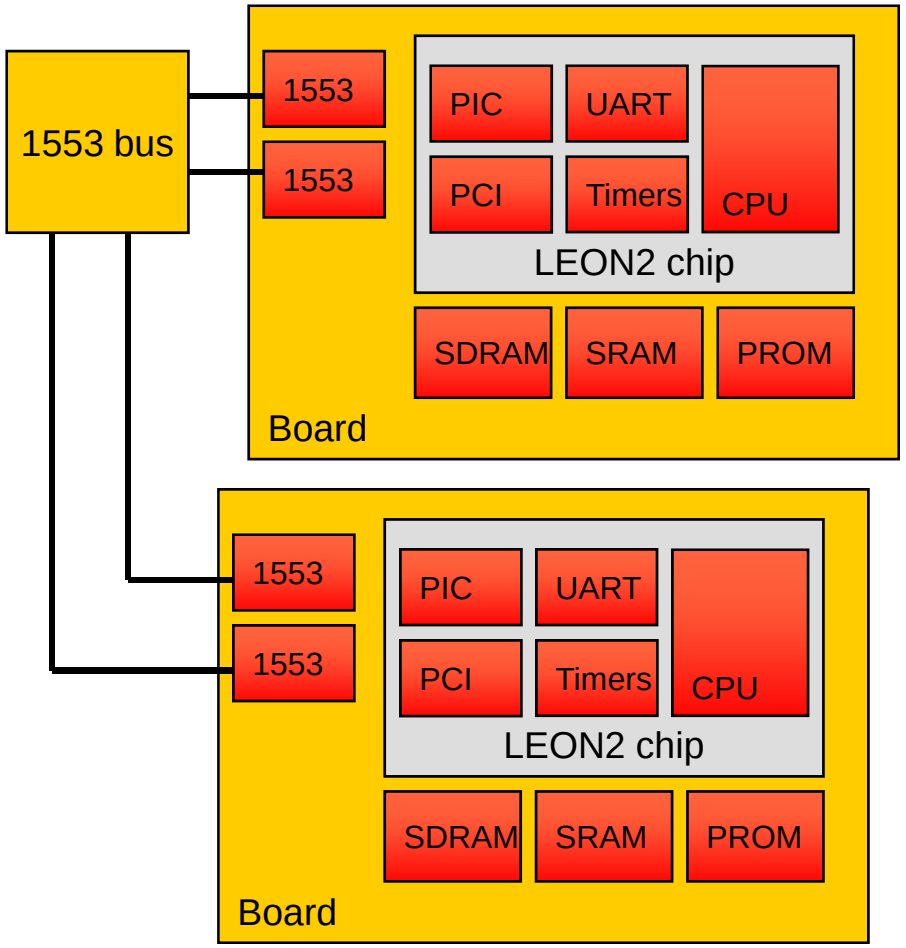
Large Target System

- Telecom Switch
  - ATM Backplane
  - Ethernet frontside
  - Serial
  - 20+ different card types
    - Control cards
    - Timer units
    - Line cards
    - Backplane switch cards
    - Multipro compute cards
    - DSP processing cards
  - 20+ cards in a rack
    - Combined arbitrarily
- Extreme system size
  - 10-100s processors
  - 10+ of GB target RAM
  - Multiple processor types
    - PowerPC 403, 405, 440
    - PowerPC 750, 750fx, 750gx
    - PowerPC 8641D, 85xx
    - PowerQUICC II 8260, 8270, 8280
    - PowerQuicc II Pro 8360
    - TI C64, C64+ DSP

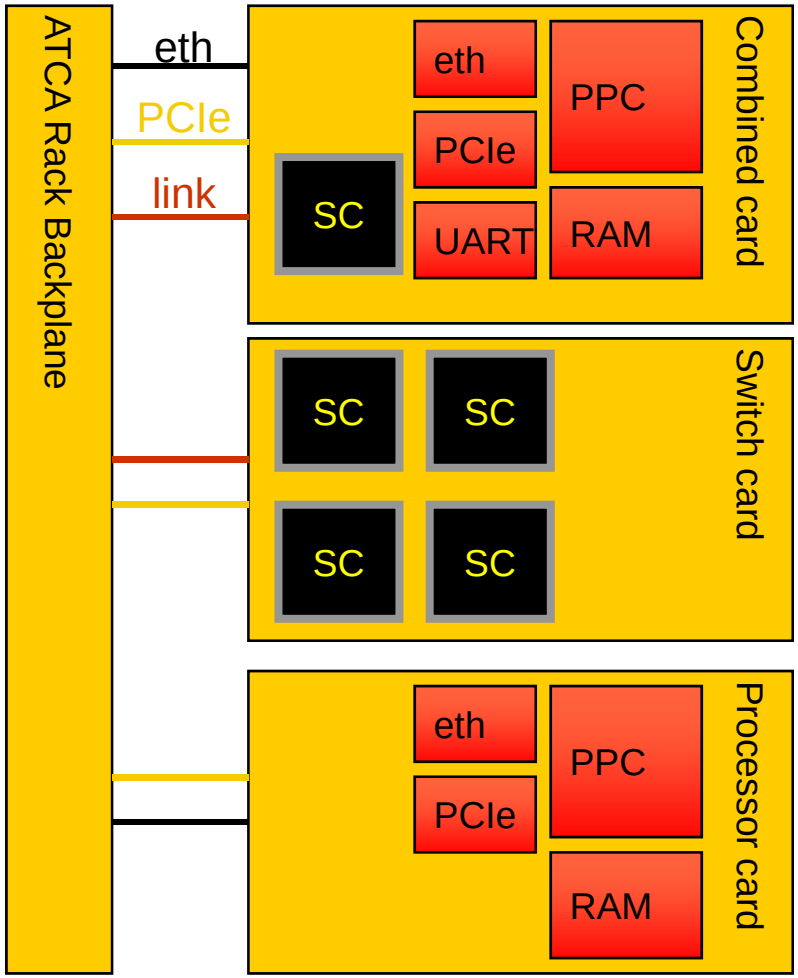


LEON Space Computer Board

- LEON2 Processor
  - Core complex
  - Memory controller
  - Serial
  - PCI
  - Interrupts
  - Clock
- Mil-Std 1553 bus
- Multiple 1553 controllers
- Multiple cards
- RTEMS OS



Virtual Reference Kit for Switch Chip



- Standard components from VT
  - ATCA rack
  - PPC 8548
  - PCIe & Ethernet traffic
- Customer work
  - Custom switch chip models
  - Custom backplane link
- Simics integration
  - Wrapping customer models into Simics models
- Features
  - Multiple cards, multiple chips
  - Same SW as real platform
  - Months before hardware
  - Shipped to subcontractors and OEMs

Debugging with Virtual Platforms

virtutech

Three Steps of Debugging

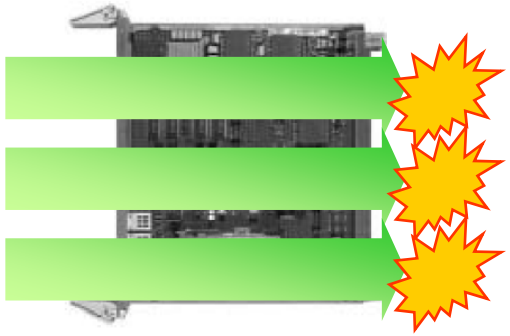
1. Provoking errors
   - Forcing the system to a state where things break
2. Reproducing errors
   - Recreating a provoked error reliably
3. Locating the source of errors
   - Investigating the program flow and data
   - Depends on success in reproduction

A simulator can help with all three steps

Simics Debugging Features

Synchronous stop for entire system

Determinism and repeatability

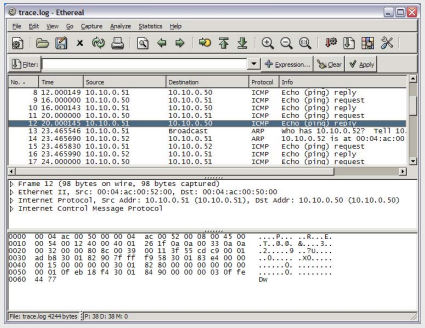


Reverse execution

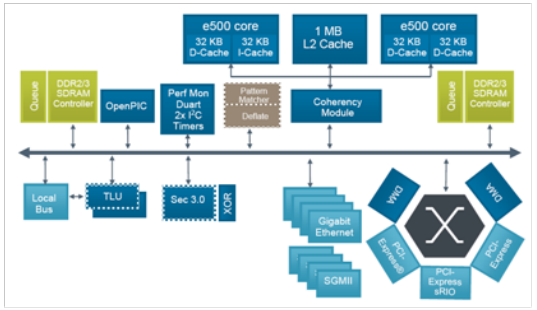
Unlimited and powerful breakpoints

```
break -x 0x0000->0x1F00

break-io uart0

break-exception int13
```

Trace anything

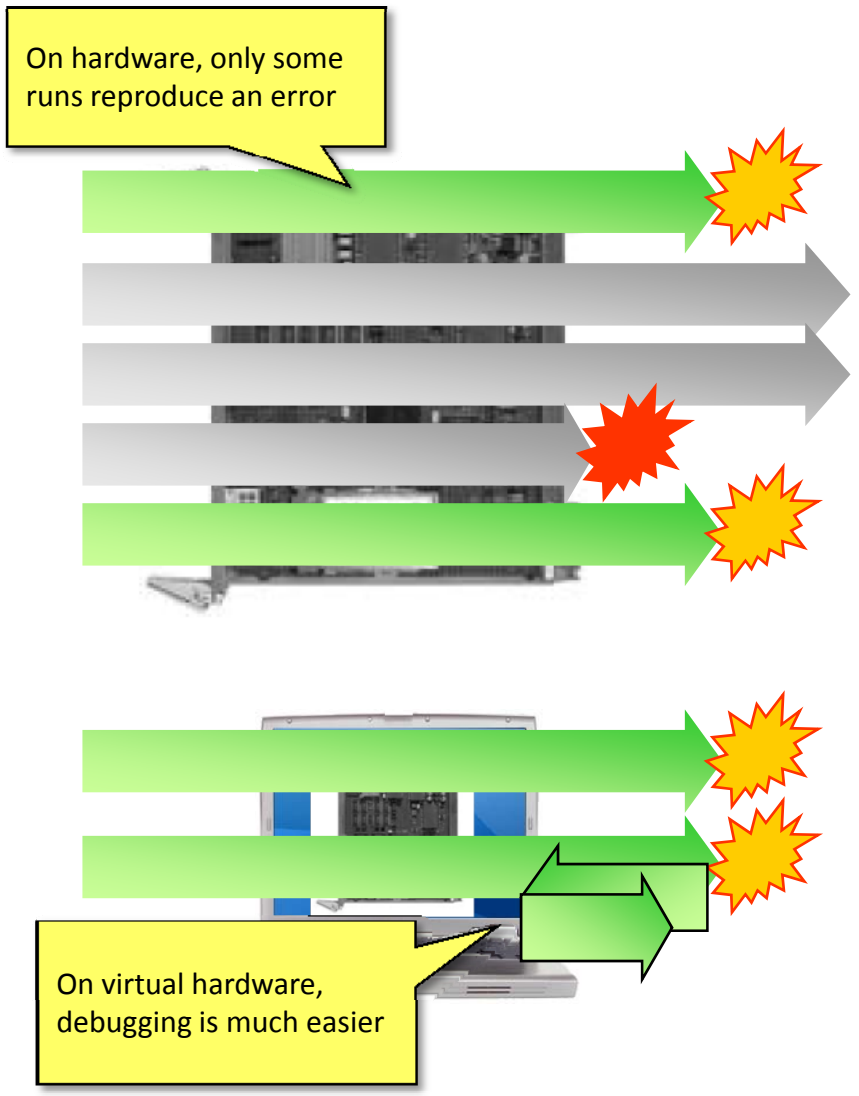


Insight into all devices



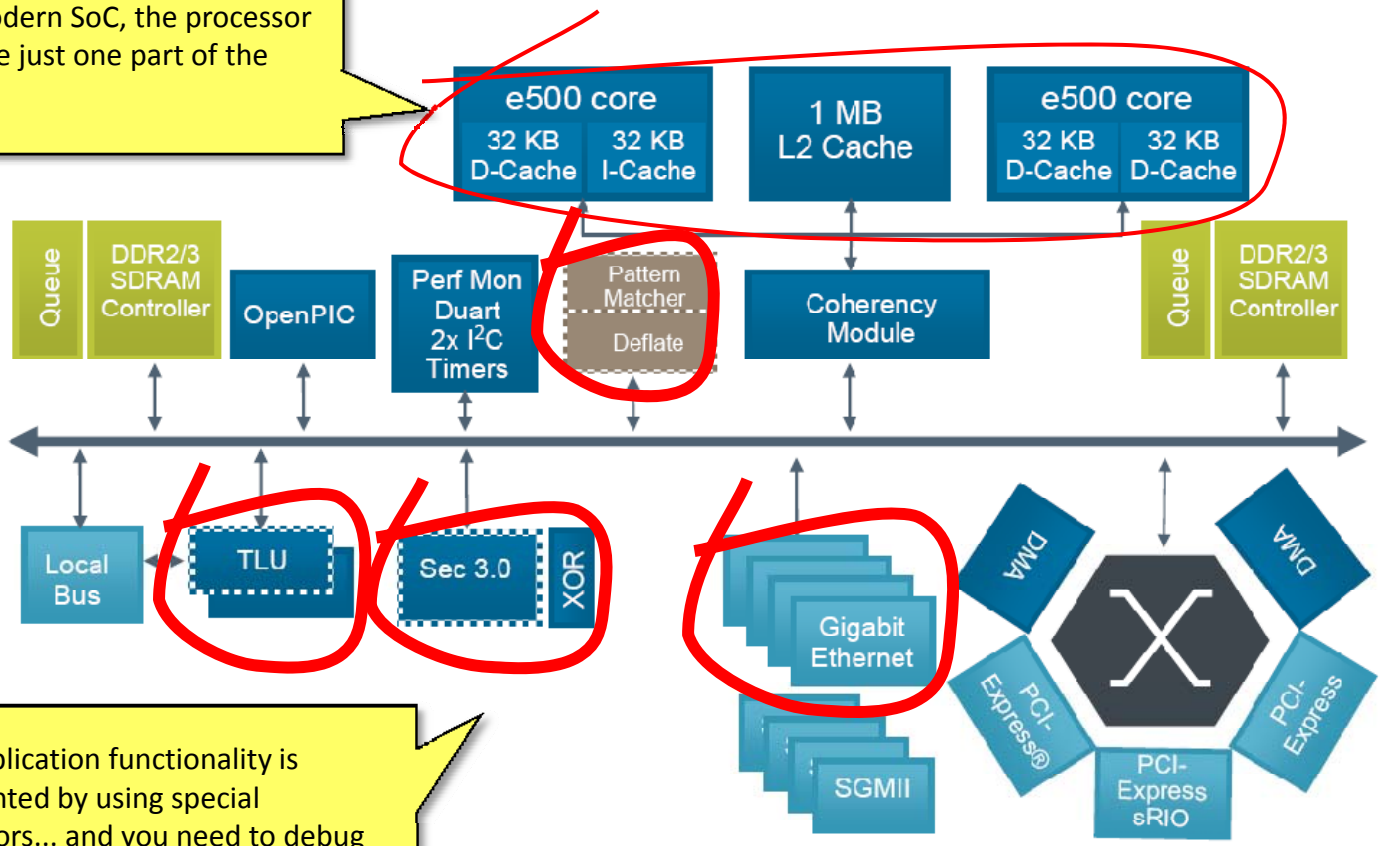
Repeatability and Reverse Debugging

- Repeat any run trivially
  - No need to rerun and hope for bug to reoccur
- Stop & go back in time
  - Instead of rerunning program from start
  - Breakpoints & watchpoints backwards in time
  - Investigate exactly what happened this time
- This control and reliable repeatability is very powerful for parallel code!



Code is not just about CPUs

On a modern SoC, the processor cores are just one part of the system



Much application functionality is implemented by using special accelerators... and you need to debug their interaction with the processors & software

virtutech

Divide-by-zero in OS Kernel

- Operating-system kernel crash in virtual model
  - Divide-by-zero right in the kernel
  - Algorithm to determine and compensate for clock skew
  - Division by difference in time between two processors
- Virtual model had zero clock skew = provoked error
  - Could have happened on a real system
  - Just not very likely
  - Typical rare problem in the field
  - Essentially testing a rare corner case in system state

virtutech

Race Condition in Serial Driver

- The problem:
  - Dual-core MPC8641D machine
  - Changed clock frequency from 800 to 833 Mhz
  - OS froze on startup – quite unexpectedly
- Investigation:
  - Only happened at 832.9 to 833.3 MHz
  - Determinism: 100% reproduction of error trivial
  - Time control: single-step code feasible
  - Insight: look at complete system state, log interrupts, check the call stack at the point of the freeze, check lock state
- What we found:
  - An interrupt service routine attempted to take a lock, before re-enabling interrupts. In the case that froze, the lock was already taken when the service routine was entered, and with no interrupts enabled there was no way for it to be released.

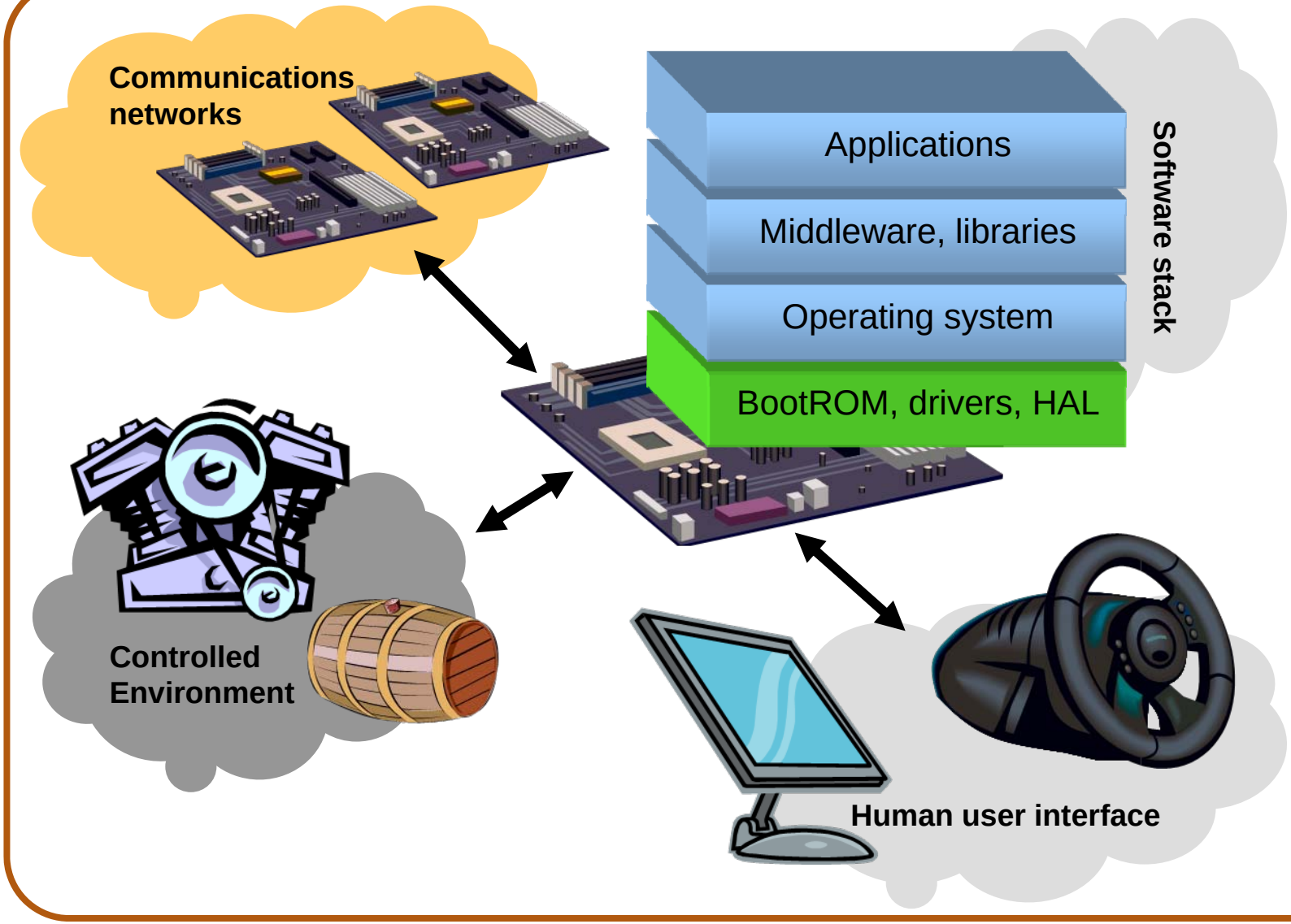
virtutech

Simulation Technology

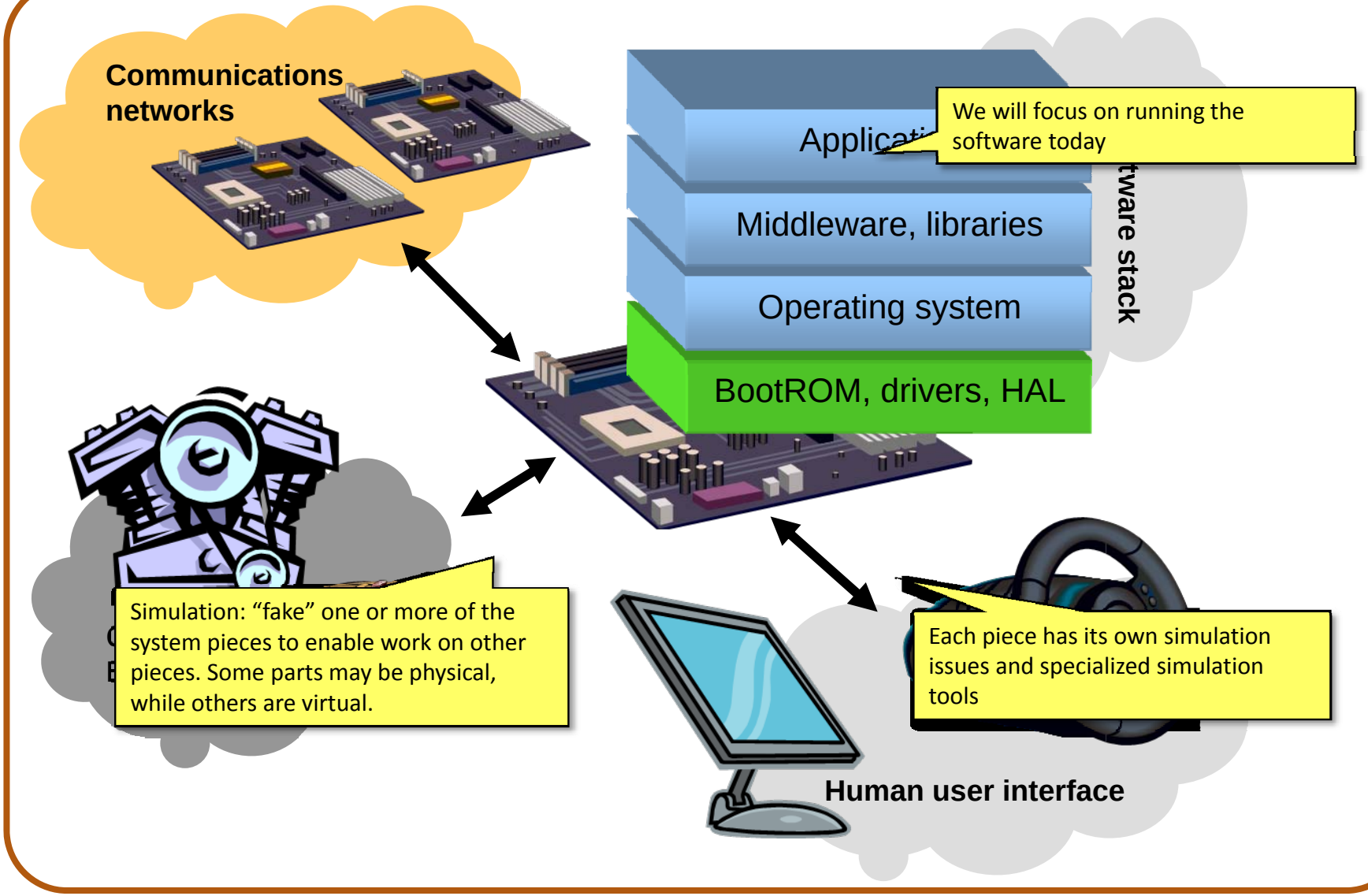
Part II…

virtutech

Embedded Computer System



Simulating Embedded Computer System



User Interface Simulation

- A category of tools of its own
- Part of many other simulation tools

- Many different levels:
  - Virtual screen & mouse like VmWare and its ilk
  - Clickable simulation of touch screens
  - Clickable panels of buttons
  - Graphics displays, text displays, LEDs, etc.
  - Full hardware mockups connected over CAN bus to PC
- Software:
  - Simulated by scripts
  - Special code for special API
  - Actual target code in some form of other simulator

Environment Simulation

- Large field for powerful commercial tools
  - MatLab/Simulink
  - LabView/Matrixx
  - MSC software
  - .. and many more ...
- In-house models common

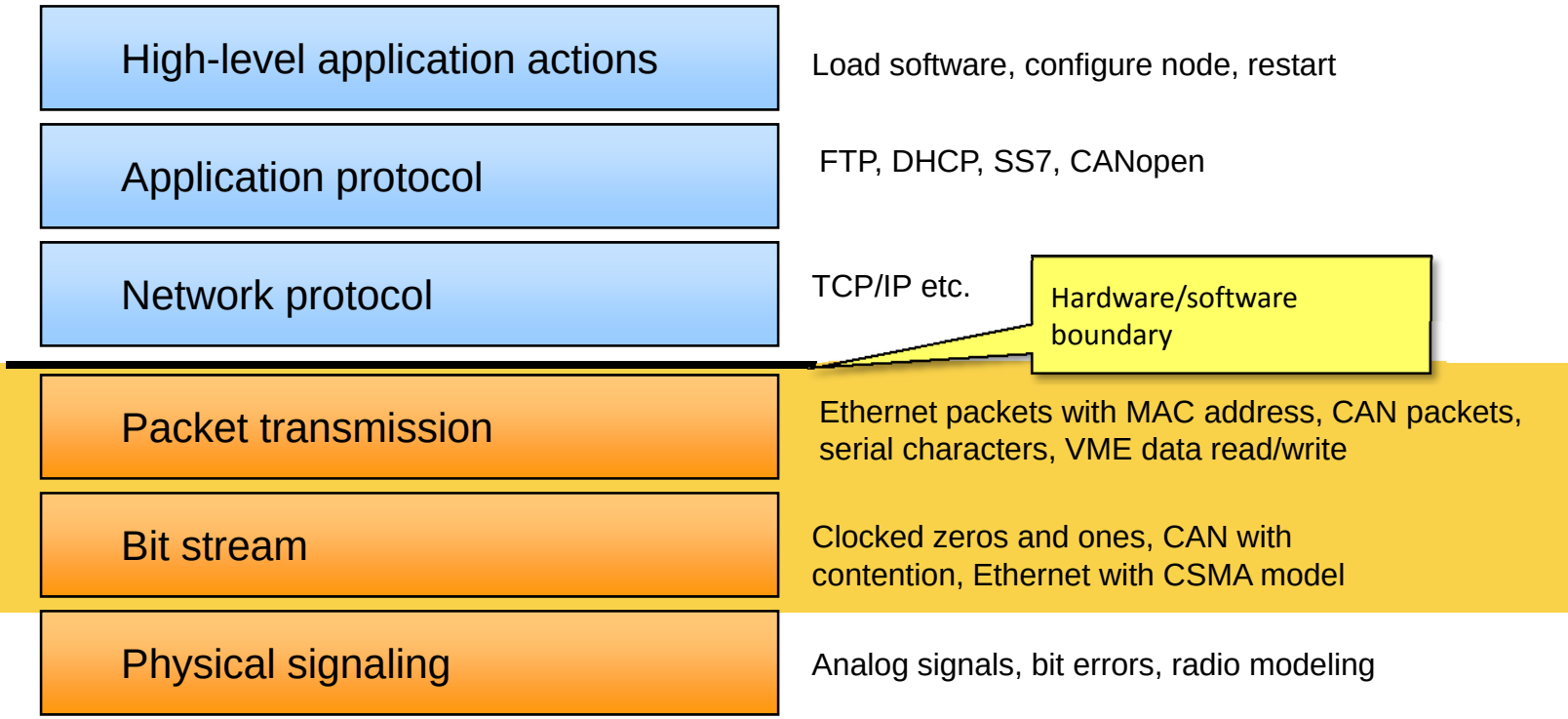
- Everybody is using it, CAD has been doing mechanical simulations for 50 years
- Commonly used for control algorithm development
- Key part of the model-driven architecture/model-driven design paradigm
- Interface to board simulation:
  - AD, DA converters
  - Digital inputs & outputs

virtutech

Network Simulation Variants

- Connections between abstracted nodes, to study communication patterns
  - Contains models of node behavior, no actual code
- "Rest of network simulation" to provide the environment for a single node
  - Generates "real" traffic
  - Implements actual protocols
  - Bidirectional reactive traffic
- Dumb traffic generation
  - Generate traffic from rules
  - Unidirectional
- Virtual packet-level network links between simulated nodes
  - No protocol understanding
  - Nodes run network stacks
- Connect physical and simulated nodes
  - Virtual machines visible on physical network
- *Network types:*
  - *Ethernet, AFDX, CAN, LIN, FlexRay, MOST, PCIe, I2C, LonWorks, ARINC 429, MIL-STD-1553, Serial, RapidIO, VME, SpaceWire, USB, FireWire, ...*

Network Simulation Levels



Computer System Simulation Technology

System Simulation use Cases

- **System-on-Chip Design**
  - Focus on hardware designer needs
  - Architecture exploration
  - Sizing, performance, optimization of hardware
- Fidelity to target is primary driver for models
  - Timing
  - Bandwidth
  - Latency
  - Bus structure
- All components are equals

- **Software Development**
  - Focus on software developer needs
  - Execute large workloads
  - Debug code
- Speed of execution is the primary driver for model
  - Abstract as far as possible
  - Approximate timing
- Work from the processor outwards
- Clear difference between processors and other devices

virtutech

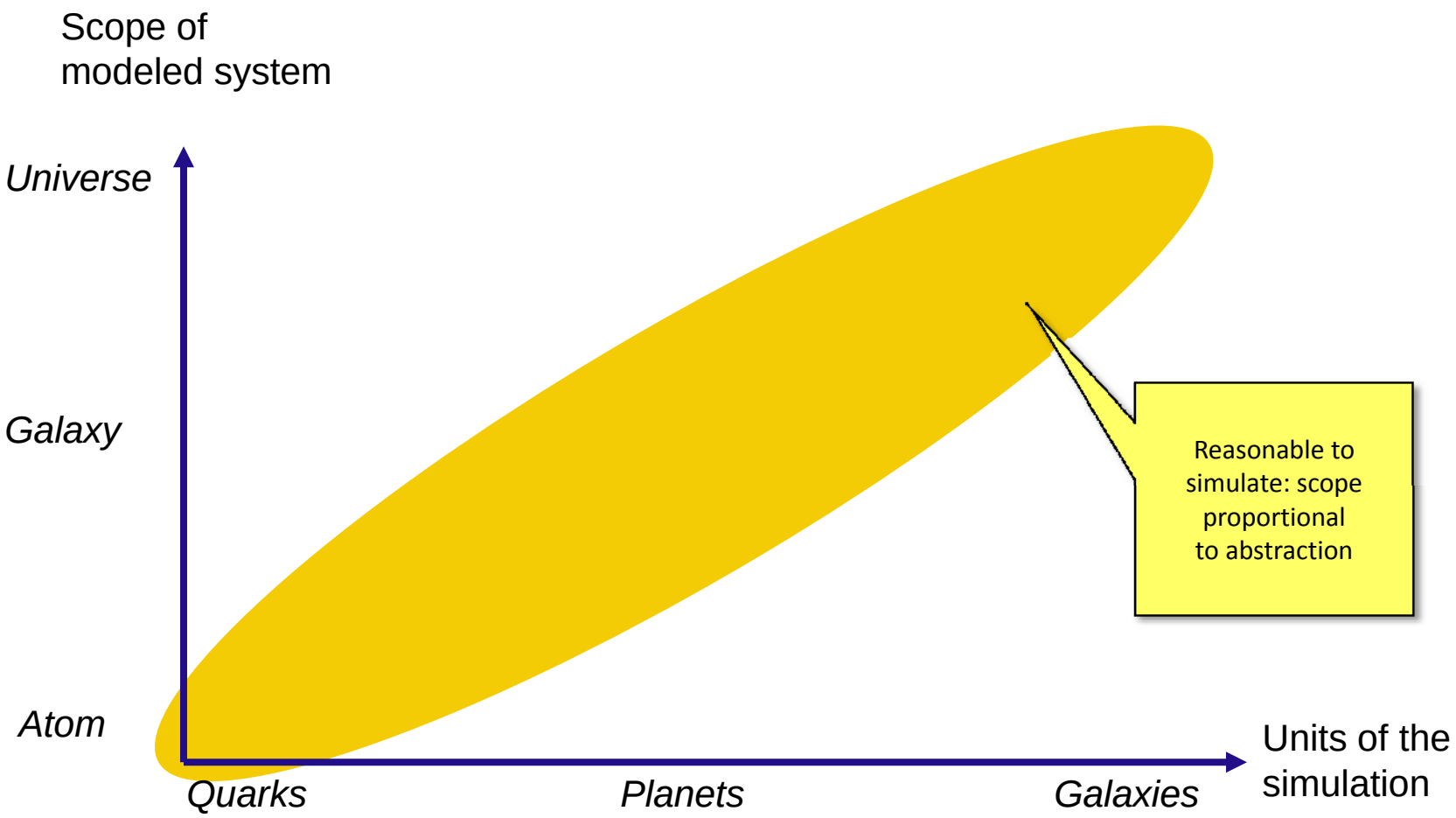
Full-System Simulation

- Detail level determines speed
  - The more detail, the slower the simulation
- Abstraction: timing precision, implementation details
- Functionality must always be correct!

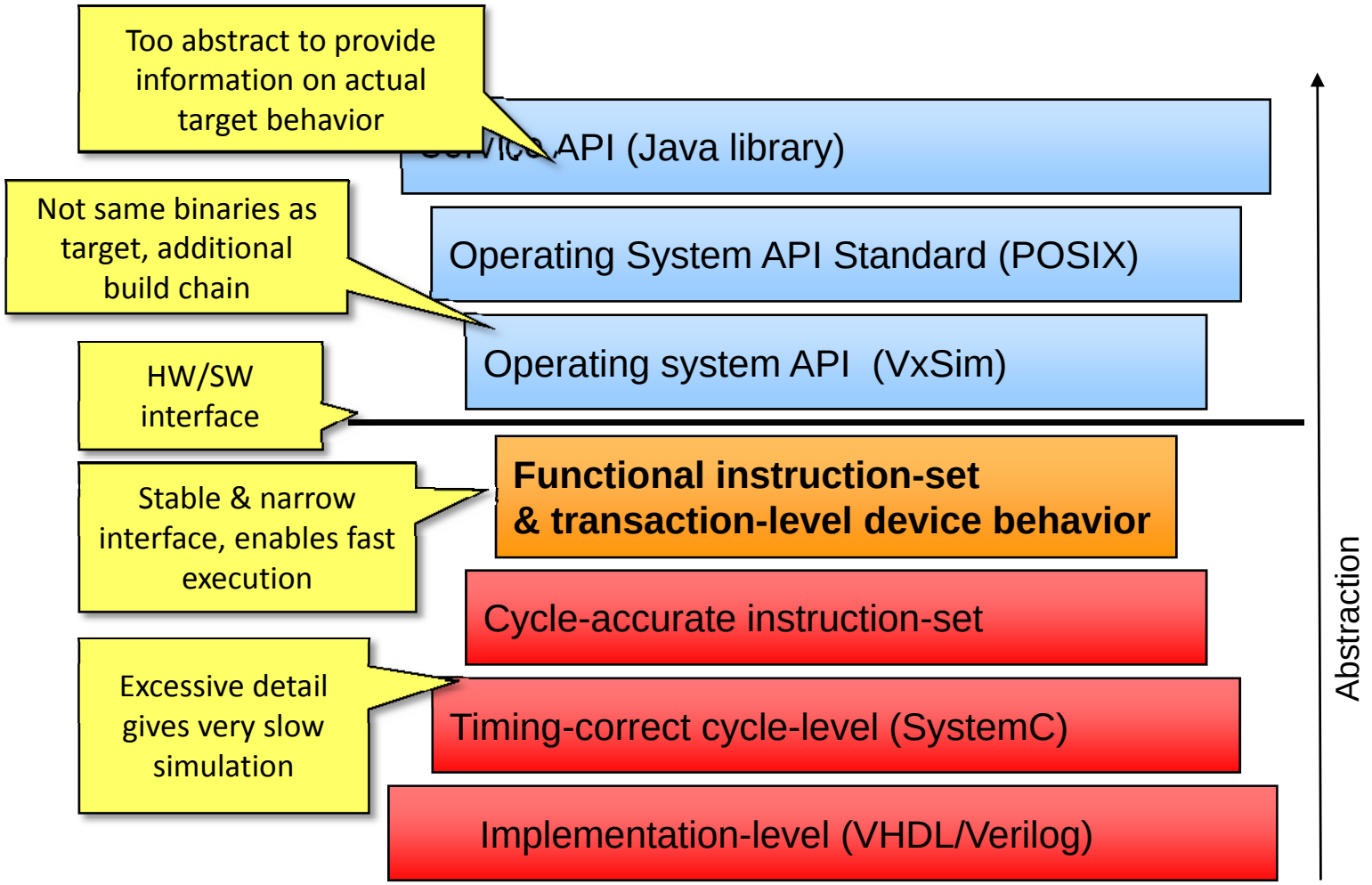
Simulation detail level	Typical slowdown	Approximate speed in “MIPS”	Time to simulate one real-world minute
Gate-level simulation	1000000	0.002	2 years
Cycle-accurate simulation	10000	0.2	7 days
Cycle-approximate simulation	500	4	8 hours
Fast functional simulation	5	400	5 minutes

virtutech

Cardinal Rule of Simulation



Abstraction Levels



Fast Functional Simulation

virtutech

The Art of Fast Simulation

- ”Know when to bluff”
  - You are in a poker game against the software ☺
- It is an art to implement just enough to fool the software, but not more
  - Details cost dev time and execution speed
  - Implement the what and not the how
  - Do work in largest possible units
    - entire Ethernet packets
    - DMA in a single step
  - ”transaction-level modeling”

virtutech

Simics Modeling Level: Processor

- Instruction-set simulation (ISS)
- **Complete** and **correct** processor functionality
  - All instructions semantics bit-correct vs real machine
  - Supervisor-mode & user-mode
  - Runs the complete target instruction set
    - Including Altivec, SSE, 3dNow, VIS, etc. extensions
  - All accessible values represented
    - User-level registers
    - Supervisor-level registers
    - Model-specific registers, ASIs, debug register, etc.
- Memory-management unit
- Timing abstracted
  - Fixed execution time per instruction
  - No cache model, normally

virtutech

Simics Modeling Level: Devices

- Hardware modeled as a **set of devices**
  - Memory map of machine (as seen by processor)
  - At the programming register level
- Model the program-visible behavior
  - Configuration registers
  - Control register
  - Data transmitted & received
- Transaction-level modeling
  - Reads, writes, DMA transfers, network packets
- Reactive, passive models
  - Only execute code when a transaction occurs
- ASICs & FPGAs
  - Model programming interface behavior
  - Not detailed implementation

virtutech

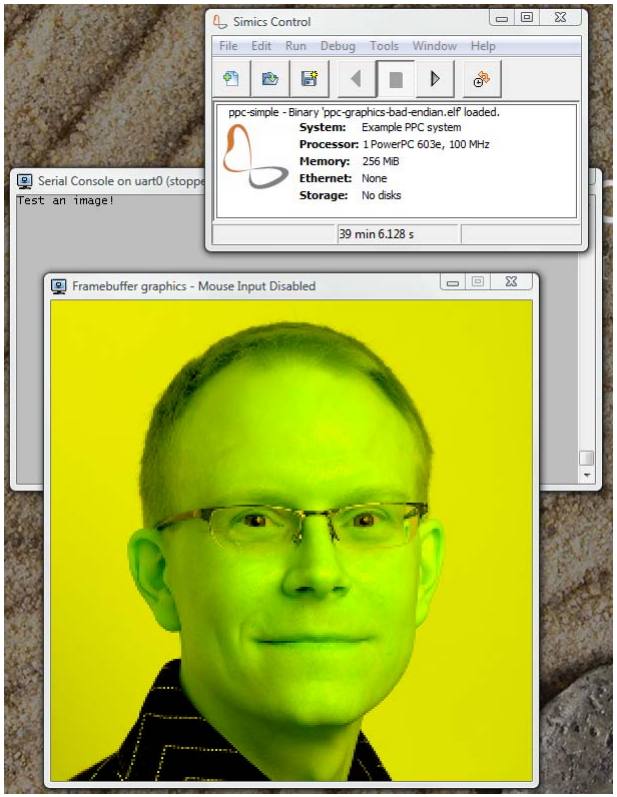
Simics Modeling Level: Networks

- Interfaced using “real” network devices
- Networks modeled at **message** level
  - Entire messages (packets, frames, ...) delivered as a unit
- Hardware addressing used
  - Ethernet MAC
  - Does not care about higher-level protocols
  - Ethernet allows IPv4, IPv6, TCP, UDP, SCP, ICMP, ...
- Any topology or addressing scheme
  - Broadcast, unicast, switched, point-to-point, etc.
- Perfect network by default
  - Introduce latencies
  - Introduce bandwidth limits
  - Introduce faults

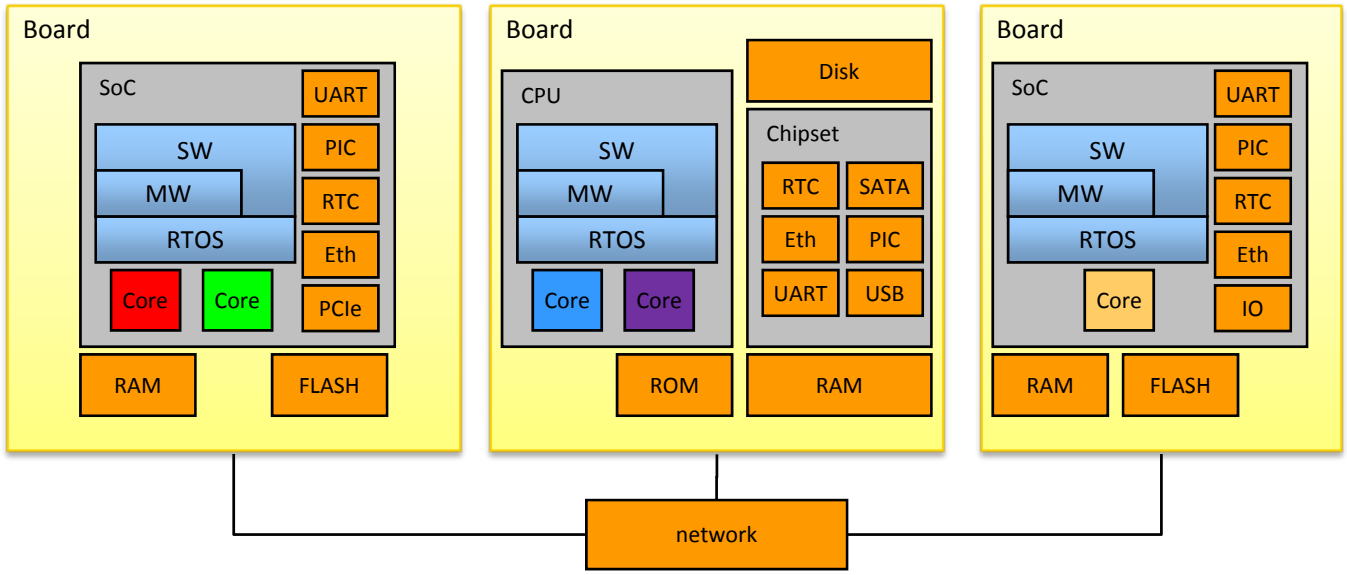
Endianness – Has to be Modeled

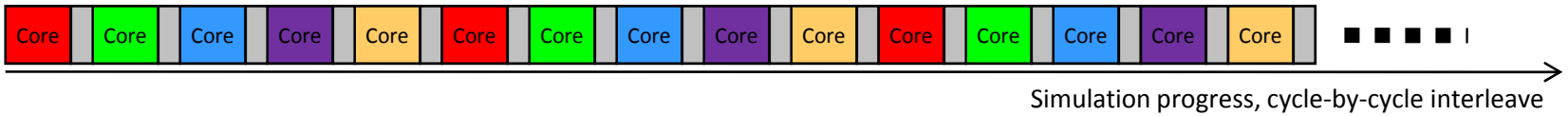
- Correct endianness

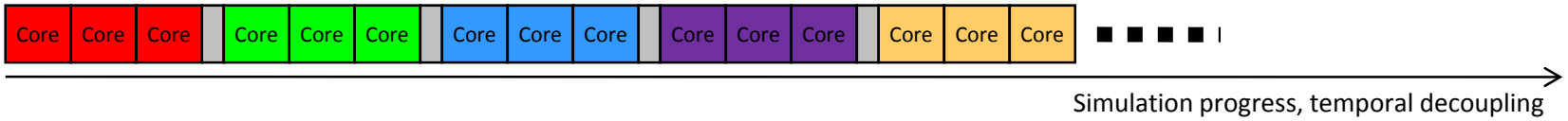
- Incorrect endianness



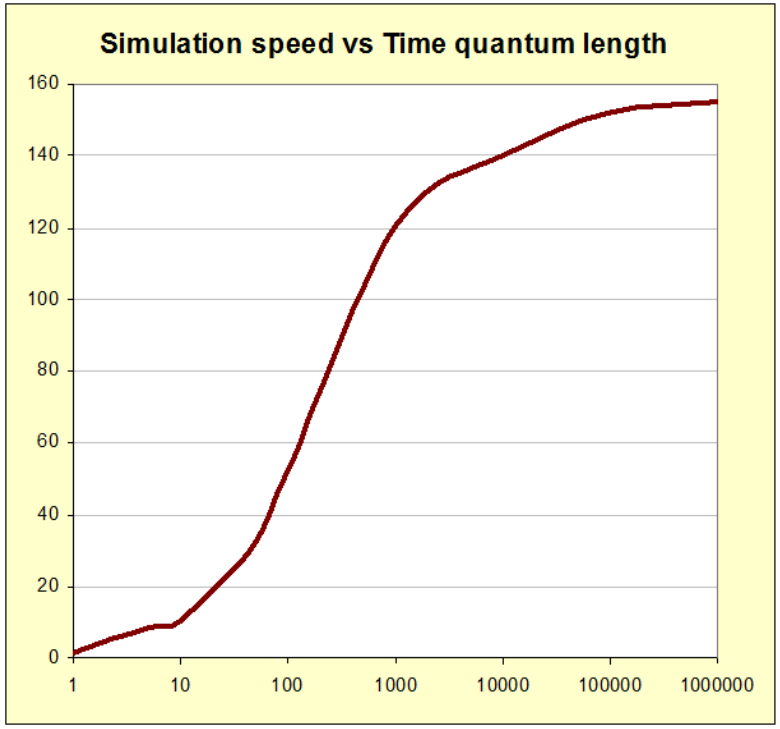
Temporal Decoupling







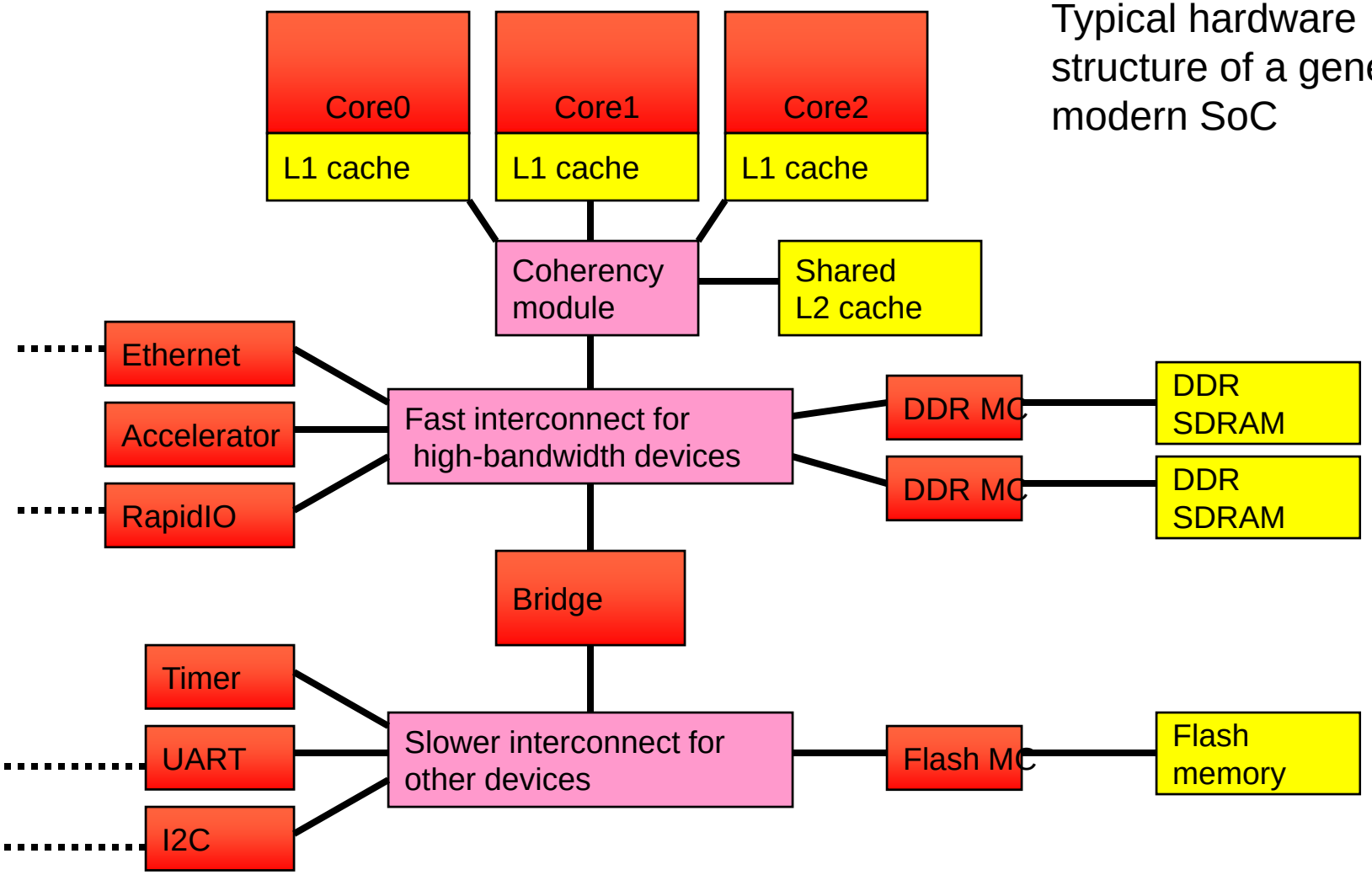
Temporal Decoupling Speed Impact



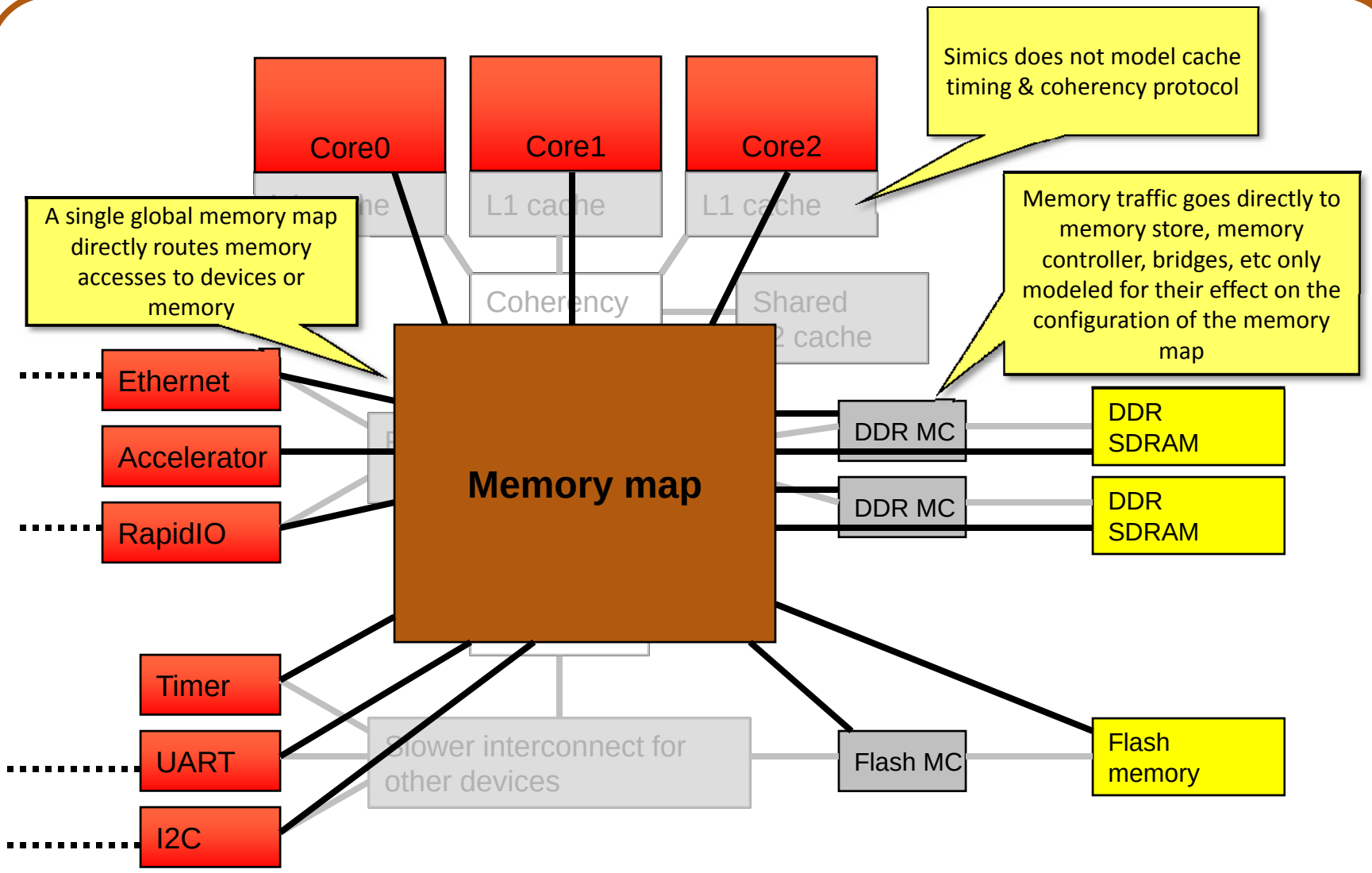
- Experimental data
  - 4 virtual PPC440 boards
  - Booting Linux
    - Which is a particularly hard workload, lots of device accesses
  - Execution quanta of 1, 10, 100, ... 1000_000 cycles
- Notable points:
  - 10x performance increase from 10 to 1000 quantum
  - +30% from 1000 to 1000_000 quantum

Memory Bus Modeling

Typical hardware structure of a generic modern SoC



Fast Memory Bus Modeling



virtutech

Dummy Devices

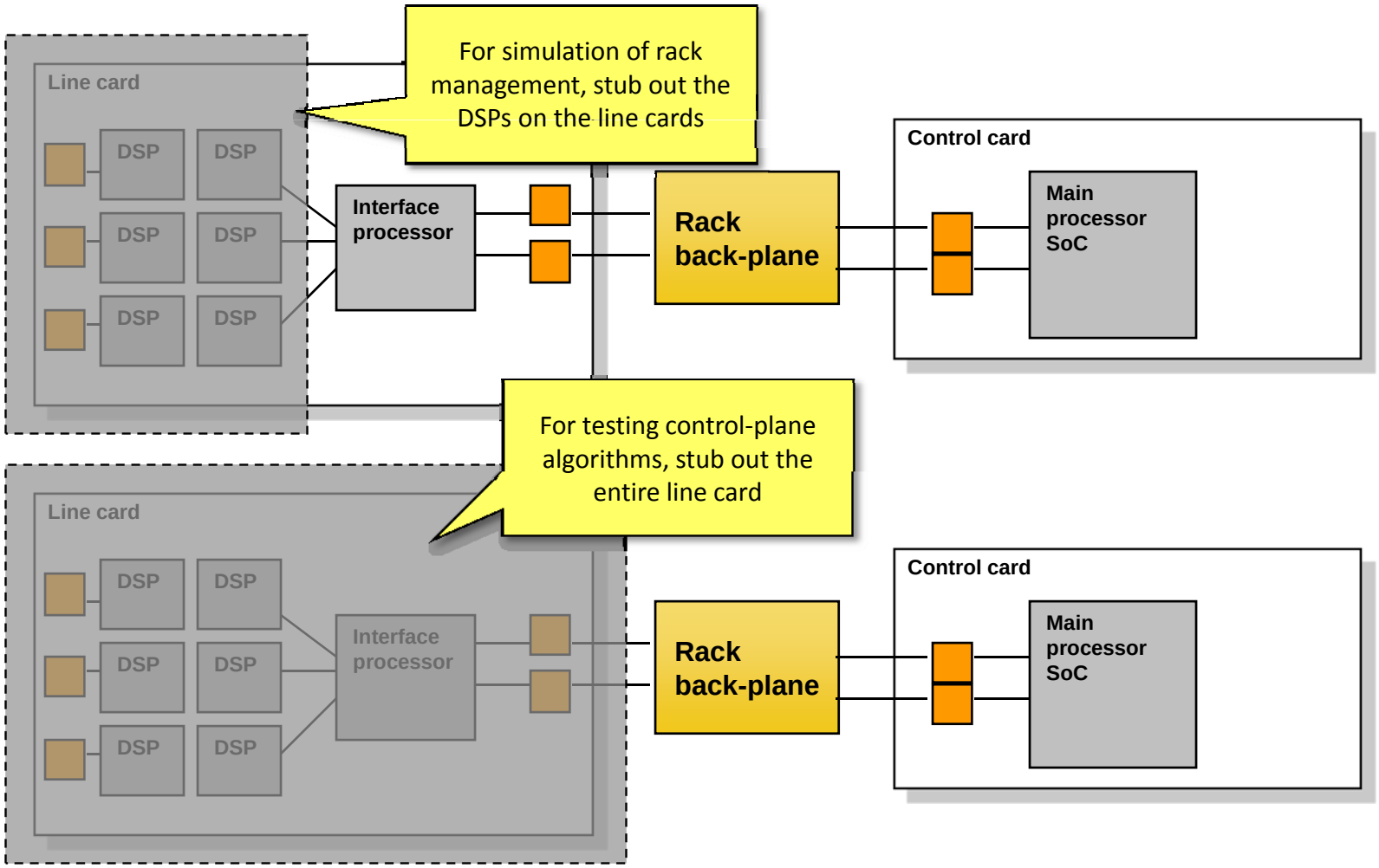
- Many devices lack interesting behavior
  - From the perspective of the software for a particular system
    - Only affect low-level system timing
    - Not used by current software setup
    - No interesting effects from using them
  - Replace with dummies that do nothing
    - But do not give “access out of memory” errors
- Examples: memory timing setup, performance counters, error detection registers, ...
  - Note that you can add them later if effects are needed

virtutech

Stubs

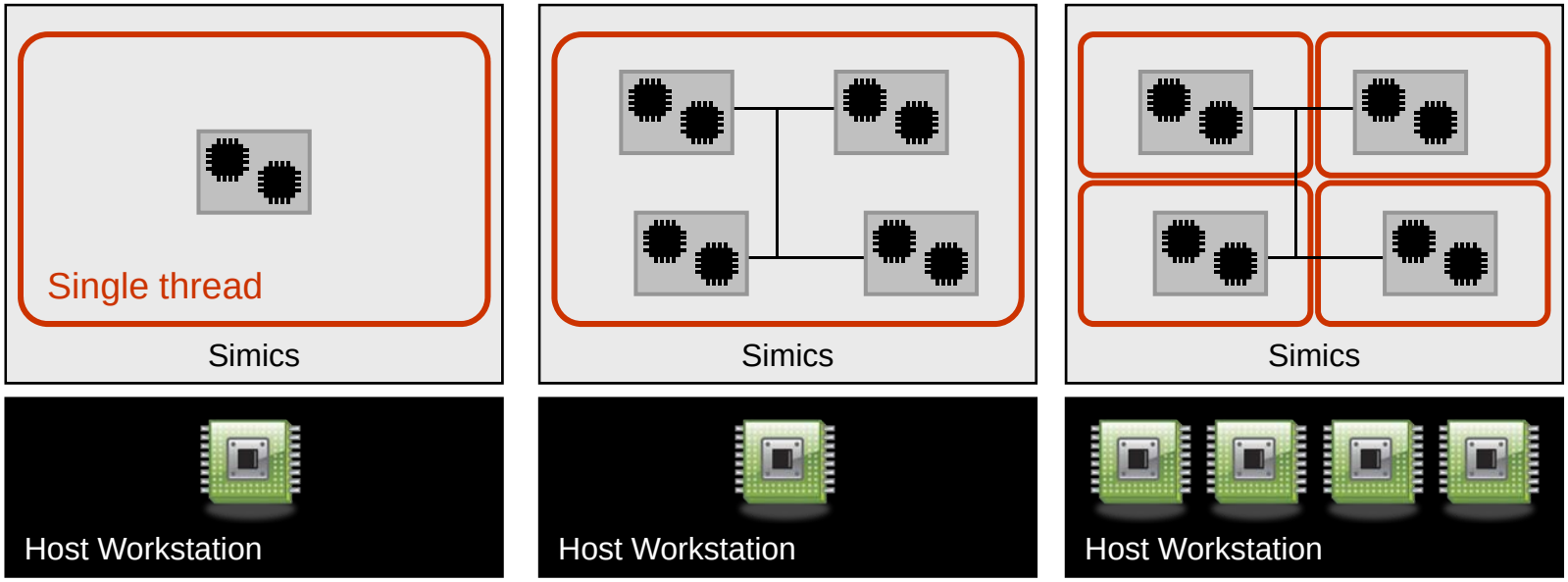
- Not all parts of a system need to be modeled
- Replace by **stubs**
  - Find an appropriate (narrow) interface to cut at
  - Replace complete model with its behavior

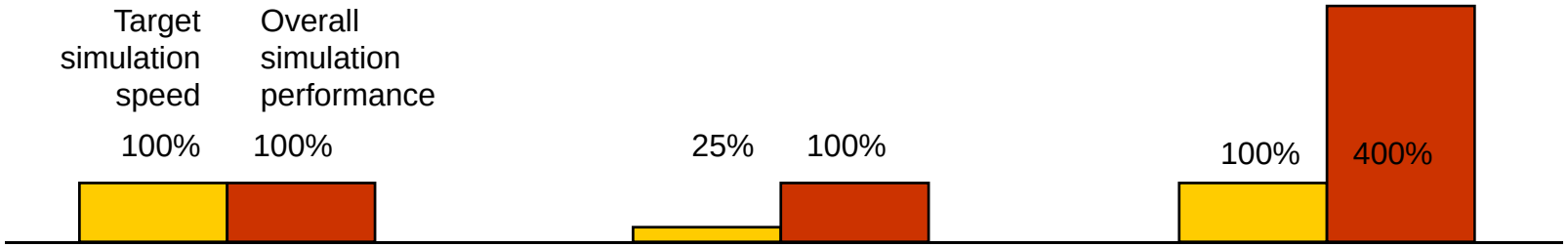
Stubs Example



Multithreading the Simulation

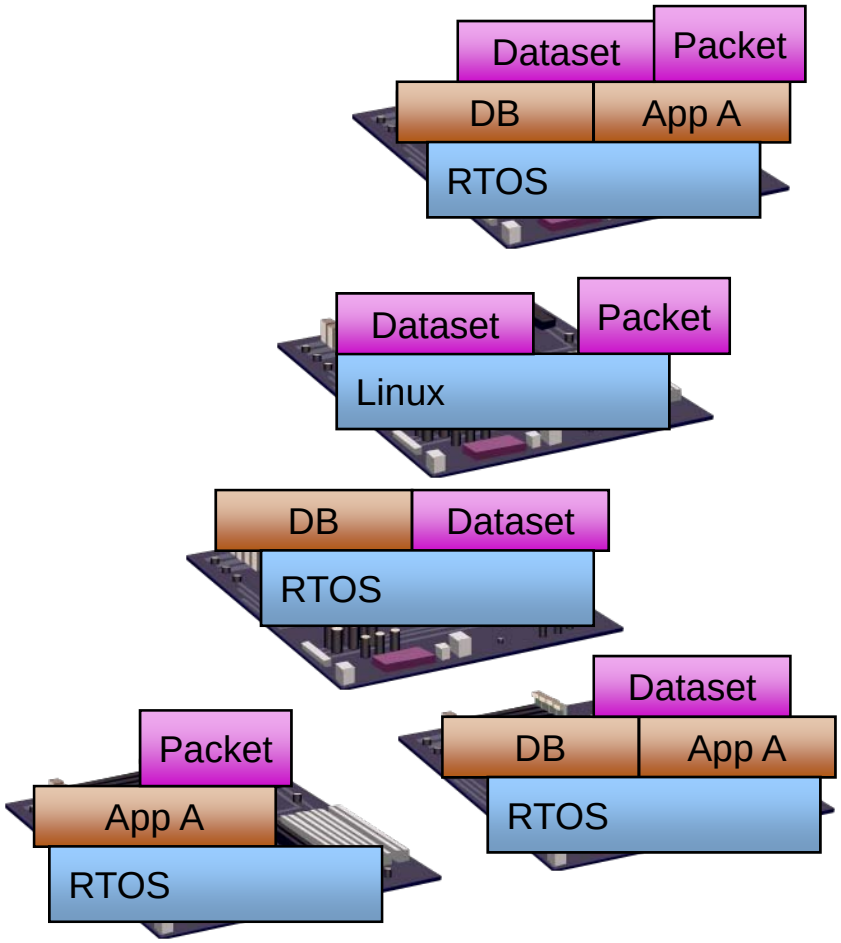
Simple system	Complex system	Complex system with Simics Accelerator



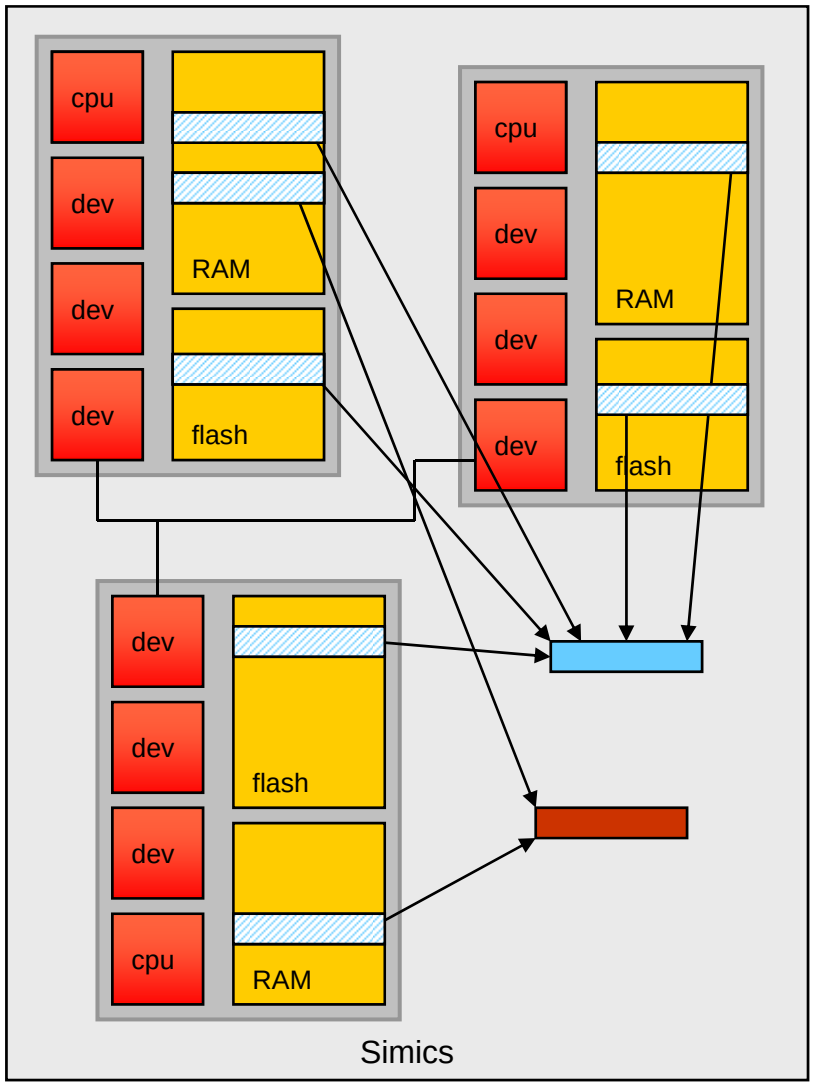


Redundancy in Target Systems

- Large systems are not built from all-unique components
- Software repeats
  - Machines use the same OS, middleware, applications
- Data repeats
  - Redundant databases
  - Data packets passed around in a cluster
- Copies within machine
  - Code and data copied from disk to memory to be used
- Simulator sees the whole system, leverage repetition to reduce memory footprint



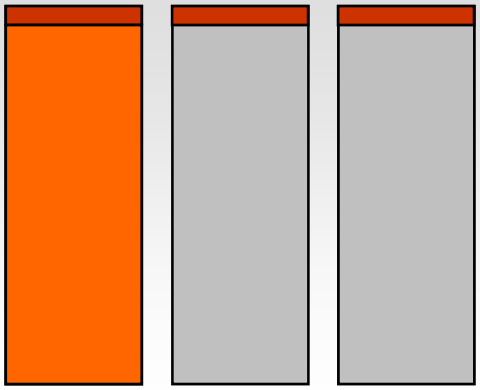
Data Page Sharing Principle



- Simics *memory images* used for all data stores (flash, ram, rom, disks, etc.)
  - Standard Simics feature
- *Identical* pages in different memory images stored in a single copy
  - Within machines
  - Between machines
  - Regardless of type of memory in the target
  - Copy-on-write semantics for safety (obviously)
- Reduces memory footprint, increase data locality, helps maintain performance

Insight: Recurring Target Data

Three 8572e/Linux machines

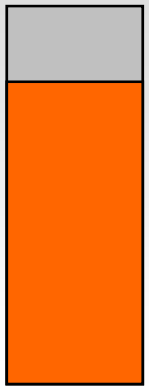


Local unique data: 4%

Shared data across machines: 96%

Total data savings: 65%

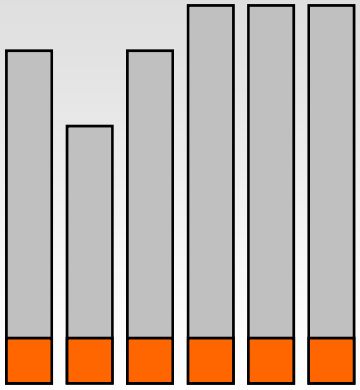
Single PPC440GP/Linux machines



Data repeated within the machine: 20%

Total data savings: 20%

Mixed network (4 mach, 6 OS)

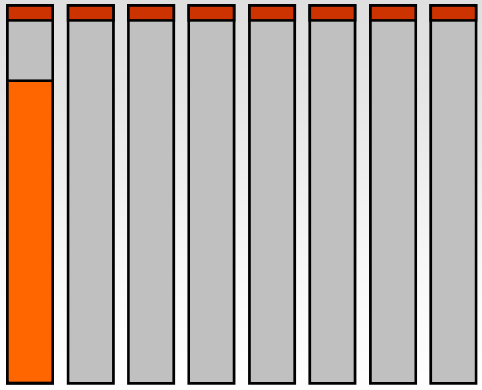


Zero pages: 90%

Other shared: 1%

Total data savings: 91%

Eight PPC440GP/Linux machines



Local unique data: 1%

Shared data across and within machines: 98%

Total data savings: 89%

All results are for networks of machines booted to prompt, but no applications loaded

Hybrid Simulation

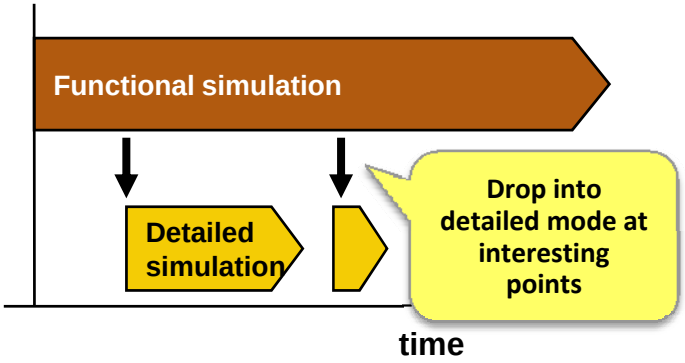
virtutech

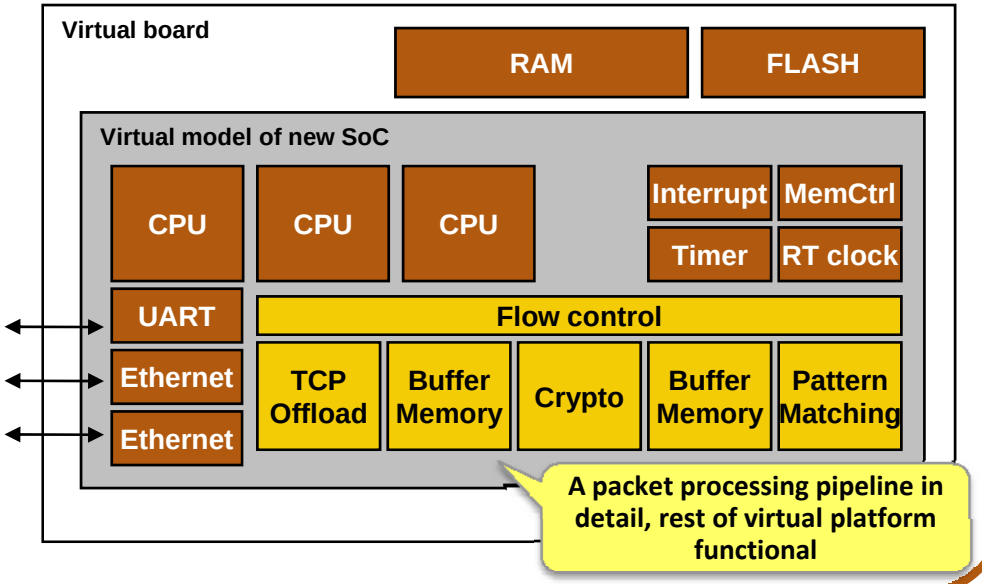
Hybrid Model

- Mix fast functional and cycle-accurate models
  - Two models of each device: fast and detailed
  - User fast simulation to get to interesting places
  - Zoom in selectively using detailed models
- Additional Simics mode: support detailed models
  - Allows out-of-order transactions and timed buses
  - Supports full bus hierarchy, not just a streamlined memory map
- First product: the Freescale QorIQ P4080
  - Functional fast models from Virtutech
  - Detailed models from Freescale (internal engineering)

Hybrid: What it Means

- Mix *temporally*: change from functional to detailed simulation when workload reaches interesting point
  - Use fast mode to position in reasonable time
- Mix *spatially*: combine fast and detailed models in the same simulation setup
  - Leverage fast models to get complete system
  - Speed up simulation by only simulating what is relevant





virtutech

Questions?

virtutech

Work for Us!

- Exjobb finnes!